Algorithm Theoretical Basis Document (ATBD) for the Conical-Scanning Microwave Imager/Sounder (CMIS) Environmental Data Records (EDRs)

Volume 16: Science Code Documentation Part 1: Purpose, Implementation, and Tutorial

Version 2.1 – 15 March 2001

Solicitation No. F04701-01-R-0500

Submitted by:
Atmospheric and Environmental Research, Inc.
131 Hartwell Avenue
Lexington, MA 02421-3126

With contributions by:
Jean-Luc Moncet, Alan Lipton, Xu Liu, Hélène Rieu-Isaacs, John Galantowicz, Ernesto Sendoya, Scott Zaccheo, Richard Lynch, Jennifer Hegarty, Sid Boukabara, Yuguang He

Prepared for:
Boeing Satellite Systems
919 Imperial Avenue
El Segundo, CA 90245

AER Document P757-TR-I-ATBD-SCDOC-1-20010315

This page intentionally left blank.

TABLE OF CONTENTS FOR VOLUME 16, PART 1

LIST OF FIGURES

LIST OF TABLES

1 PURPOSE AND SCOPE

1.1 Document

This document describes the Conical-Scanning Microwave Imager/Sounder (CMIS) software, including program structure, module definitions, and inputs/outputs. The first volume discusses the core retrieval module. The second volume lists all the include files and functions of the core retrieval module. The third volume lists all the individual EDRs. Together with the Algorithm Theoretical Basis Document (ATBD), this document serves as a guide to the software delivered by Boeing and AER to the NPOESS program.

1.2 Software

The CMIS software system is written in FORTRAN and C++ and should be compiled with a FORTRAN 90 and gcc compilers, respectively. Its performance statistics has been generated on SGI Challenge and SGI Octane machines using the IRIX 6.4 operating system. Results presented at the CMIS PDR (and documented in the EDR volumes of the ATBDs) demonstrate the performance of the programs described in this document. The software has been tested on SUN/Solaris and Linux. The software can run in a multiprocessor configuration environment, although this code release and description do not utilize the parallel processing features inherent in the algorithm. The output from the CMIS code is compatible with SIMSTAT, the government supplied statistical EDR analysis package. Data formats supported by this code include NOAA-88. Each individual EDR has been documented independently from the core retrieval module. Installation and execution instructions accompany the CD distribution of the code. The user should refer to the README files on the CD for detailed installation and building information.

1.3 System Requirements

The CMIS software has been tested on SGI, Solaris, and Linux operating systems. The build utilities automatically detects the operating systems and generates executables accordingly. The performance of the code has been conducted based on a 195MHz RS10000 processor. The top-level command uses *perl* utility. The perl library should be at least v5.005-02 or higher. GNU make utility for the makefiles of all levels should be V3.77 or higher. For the C/C++ code, gcc make utility should be 2.95.1 or higher (optional). The Fortran code is compiled using F90 7.30

(from SGI). The netCDF facility should be 3.4 or higher (provided with the CMIS software)for the I/O files.

2 SOFTWARE DESIGN AND IMPLEMENTATION

2.1 Algorithm Principles

The theoretical principles of the CMIS retrieval algorithm are described in the individual ATBDs.

2.2 Software Principles

- *Accuracy*. The input/output arguments for the modules have been carefully examined, with special attention to matrix manipulations, to achieve high accuracy for the retrieval algorithm.

- *Efficiency*. The software design is the result of a thorough study of the theoretical algorithm, the numerical operations involved in the computation of mathematical functions, and the currently available computer systems, with the purpose of obtaining the fastest speed for the software.

- *Flexibility*. The software can be configured using internal parameters or via command line options, which allows it to be adapted to other instruments or the same instrument with different configurations for retrieval or validation purposes.

- *Integrity*. A consistent naming and modularization convention is used, e.g., the microwave and infrared components have the same structure at all stages of the retrieval. This facilitates the maintenance of the software.

- *Modularity*. The software design is highly functional, with each module performing limited and well-defined tasks. This makes the code portable and compatible when individual modules are replaced with new versions.

The CMIS software contains approximately 100 functions and subroutines organized in a modular system, enabling encapsulation of computing details and the reuse of code libraries. A complete listing of the subroutines is provided in the second volume of this document (note that the subroutines are ordered alphabetically by their function names, not the file names). The subroutines and functions fall into three classes: 1. *drivers* for the main program to accomplish steps in the flow diagram, 2. *calculators*, i.e., subroutines and functions producing algorithm step answers (e.g., ***ossrad_mw*** that calculates the microwave radiances and derivatives) and 3. library *utilities*, i.e., functions and subroutines to perform generic tasks. Two sets of libraries are used: mathematical functions for matrix manipulations (matrix transpose, matrix multiplication, etc.) and memory management functions. The latter allow an efficient storage of the retrieved parameters in a single large block of allocated memory (including index offsets).

2.3 System Considerations

The capability of running the software on a multiprocessor SGI machine has been studied as well. Although the study has shown promising results, the option of using multiple processors is excluded from the current version of the software. Different optimization options for the compiler have been explored, yielding consistent results.

2.4 Top-Level Design

Figure 1 shows the flow diagram for the CMIS retrieval algorithm. It consists of five major modules (shown in rectangular boxes in Figure 1):

- *Initialization*. This module initializes model parameters and assigns appropriate I/O files.

- *Input and Pre-processing*.

- *Retrieval*. Its purpose is to obtain an accurate first guess for the joint microwave and infrared (second stage) retrieval.

- *Quality Control*.

- *Output and Post-processing*. This is the last phase of the retrieval producing the EDRs at the reporting grids.

A detailed description of the individual modules is given in Sections 8-14 of the ATBD. In Section 2.5, each module will be described from a software perspective.

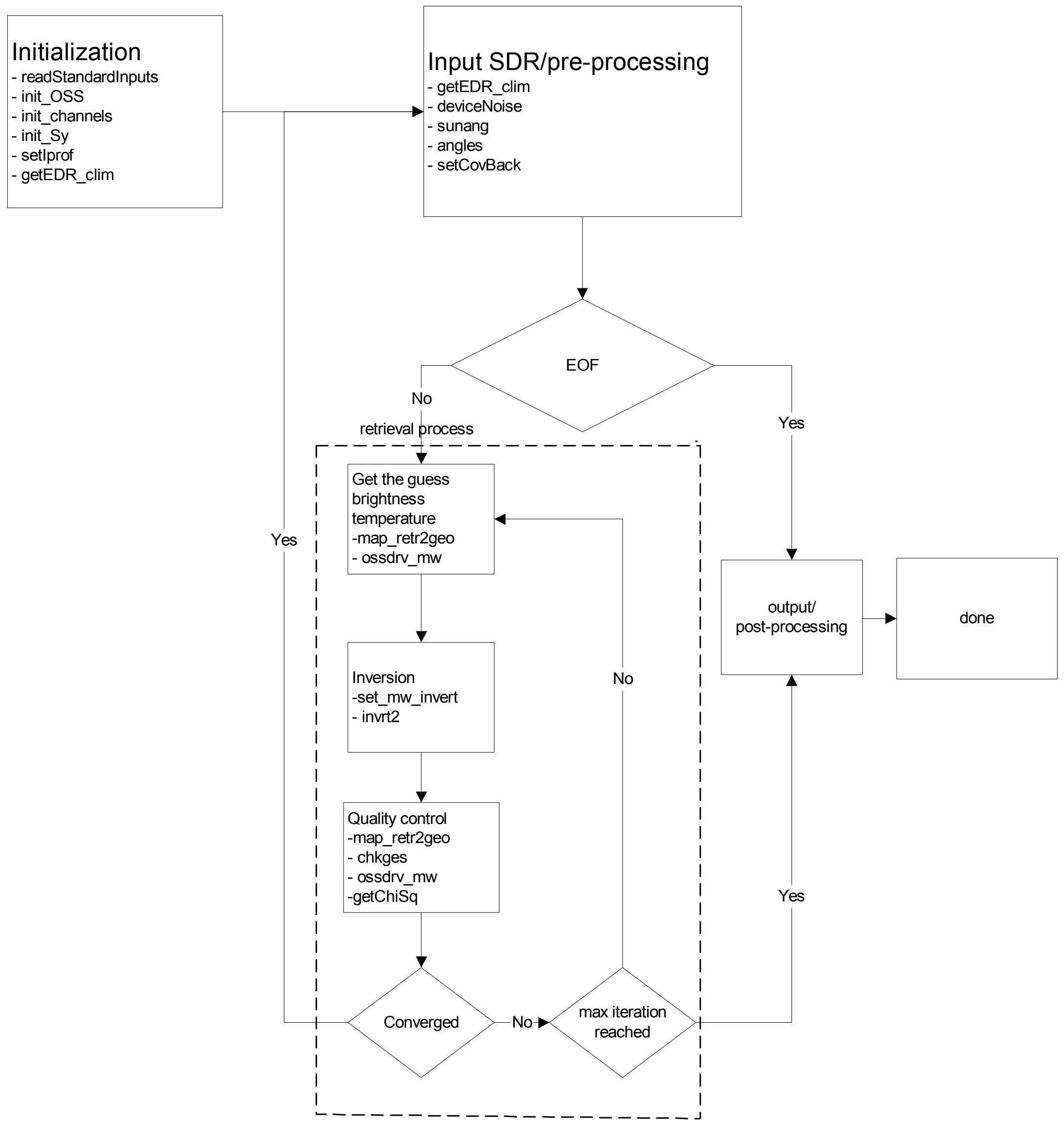

Figure 1: Top-level Flow Diagram for the CMIS Retrieval Algorithm.

2.5 Detailed Design

This section describes the individual software elements in the top-level flow diagram described in section 2.4. For each module in the top-level flow diagram (Figure 1), a decomposition into its software components (functions and subroutines) is presented in a graphical form, with each graph starting from the top-level driver for the retrieval process called ***retr.*** The driver program initializes data inputs, controls the flow of processing, and passes data between modules. The program contains internal tests to ensure a smooth execution of the algorithm, e.g., checking the consistency in the dimensions for various arrays and, before each task, assessing the quality of products from previous tasks.

In this section, in addition to their graphical presentation, functional descriptions of the functions and subroutines will be given in a tabular form for each module. In the Appendix to this document, more information about the individual software components is presented in the form of printouts of the header information from the FORTRAN files.

2.5.1 Initialization

During initialization, various static datasets (i.e., datasets that are not updated on a regular basis) are read by the algorithm. In addition to reading static input files, another task performed during initialization is the opening of the dynamic input files and the output files. A graphical presentation of the module and the functional description of its components are given in Figure 2 and Table 1, respectively. The input files whose names are specified through a control file are listed in Section 2.7 (Table 8 and Figure 8).

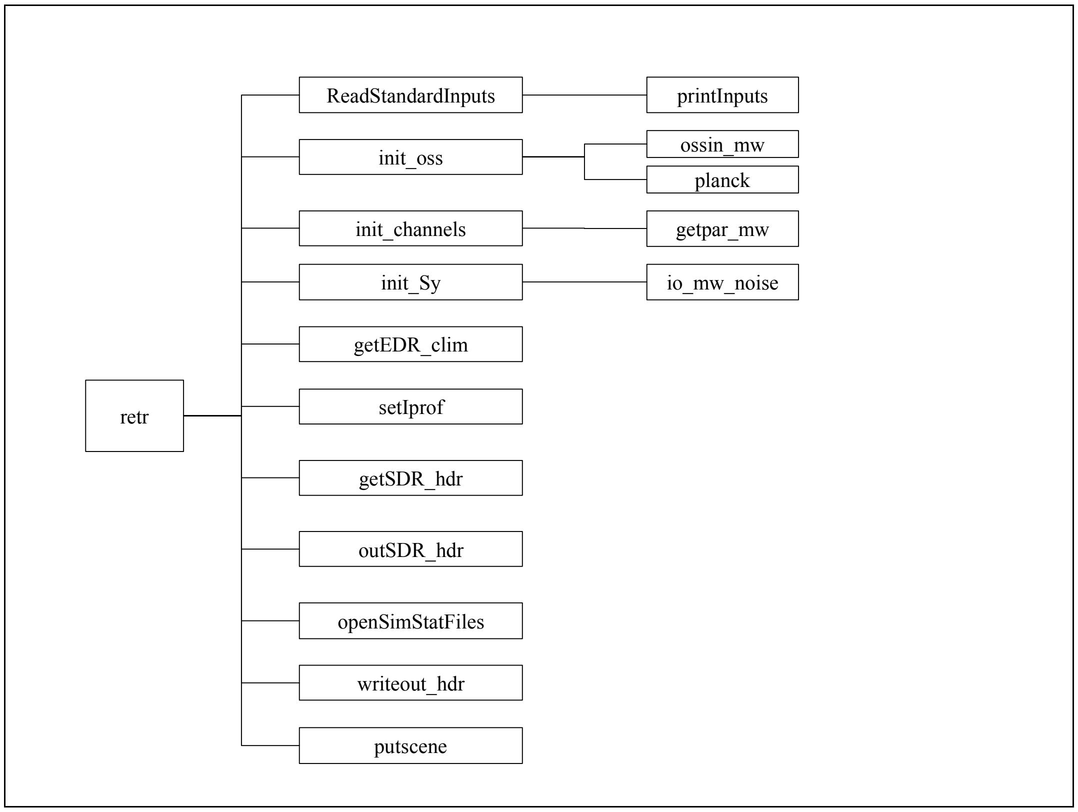

Figure 2: Software Components of Initialization.

Table 1: Functional Description of Initialization.

1st level	**2nd level**	**description**
ReadStandardInputs		Loads parameters from the namelist file
	printInputs	Prints standard inputs on screen
init_OSS		Initializes OSS model
	ossin_mw	Loads optical depth table for OSS MW model
	planck	Single-precision planck function
init_Sy		Initialization of noise and error
	io_mw_noise	Inputs MW noise vector
getEDR_clim		Loads background and covariance
setIprof		Sets up flags for trace gas retrieval
getSDR_hdr		Reads the main header of a SDR file
outSDR_hdr		Writes out SDR header
openSimStatFiles		Opens files in IPO format
writeout_hdr		Writes IPO-formatted header to the opened files
putscene (open_mode)		Opens netCDF formatted files
	handle_err	Exits in case of error
	ShiftNcdRec	Shifts pointer by 1 record

2.5.2 Input and Pre-processing

The SDR input files opened during initialization are read. Surface type determination is performed using MW radiances and the land mask. Based on the surface type, the appropriate background profile and covariance matrix is adopted. Surface pressure is computed using NWP fields. Unlike other input files, which can be assigned dynamically via the control file, the file names containing NWP fields are specified within the FORTRAN code. The noise is currently estimated to the best of the current knowledge. However, in the future it will be provided as part of the SDRs. The structure of the input and pre-processing module is shown in a graphical form in Figure 3 and the functional description of its individual components is given in Table 2.

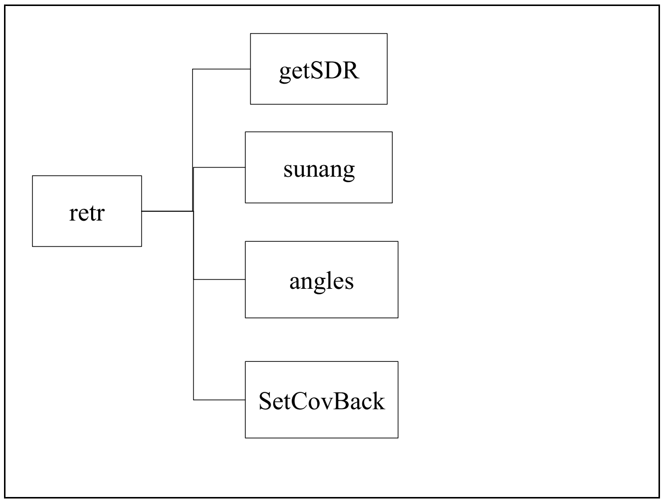

Figure 3: Software Components of Input and Pre-processing.

Table 2: Functional Description of Input and Pre-processing

1st level	Description
getSDR	Reads each record of SDR
sunang	Calculates solar zenith angle
angles	Calculates satellite view angle
SetCovBack	Assigns background and its covariance

2.5.3 Retrieval

A description of the retrieval algorithms is given in the ATBDs. The maximum iteration number is input as one of the control parameters in the configuration file. The functions and subroutines utilized by this module are shown in Figure 4 and their description is given in Table 3.

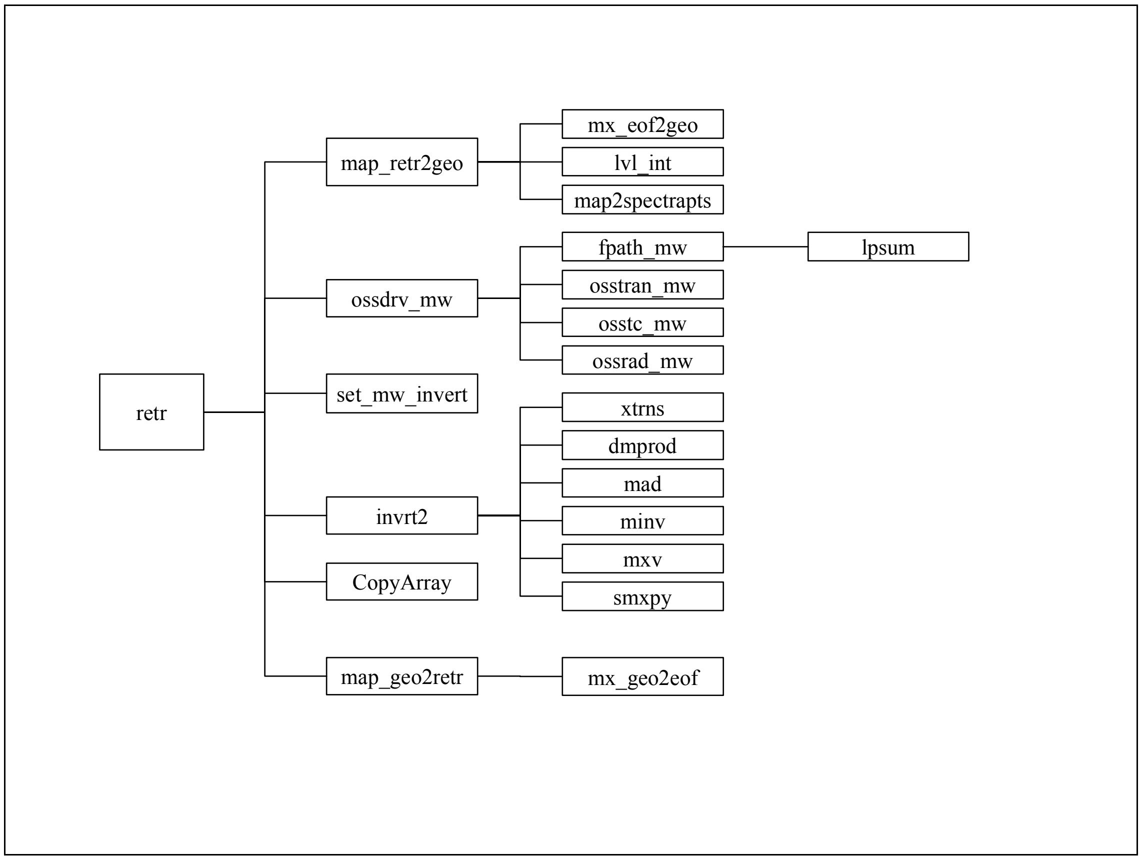

Figure 4: Software Components of Retrieval.

Table 3: Functional Description of Retrieval.

1st level	2nd level	3rd level	Description
map_retr2geo			Converts the retrieval parameters from the retrieval to the geophysical domain
	mx_eof2geo		Converts matrix from geophysical to retrieval domain
	lvl_int		Computes the surface value via interpolation
ossdrv_mw			caller of OSS forward model in MW band
	fpath_mw		Calculates MW layer-average temperature and molecular amount
		lpsum	logarithmic pressure interpolation
	osstran_mw		Computes layer optical depths for molecular species in the MW band
	osstc_mw		Calculates optical depths for liquid clouds
	ossrad_mw		Calculates MW radiance and derivatives
set_Mw_Invert			Converts derivatives from geophysical to retrieval domain in the MW band
invrt2			Performs profile inversion
	xtrns		matrix transpose
	dmprod		matrix multiplication
	mad		Addition of a matrix and a vector
	minv		double-precision matrix inversion
	mxv		product of a matrix and a vector
	smxpy		product of a vector and a matrix
CopyArray			Copies an array
map_geo2retr			Converts parameters from the geophysical to the retrieval domain
	mx_geo2eof		Converts matrix from geophysical to retrieval domain

2.5.4 Quality Control

This module is presented in Figure 5 and Table 4. Its fundamental purpose is to guarantee the physical integrity of each retrieval parameter during the process and its retrieval quality.

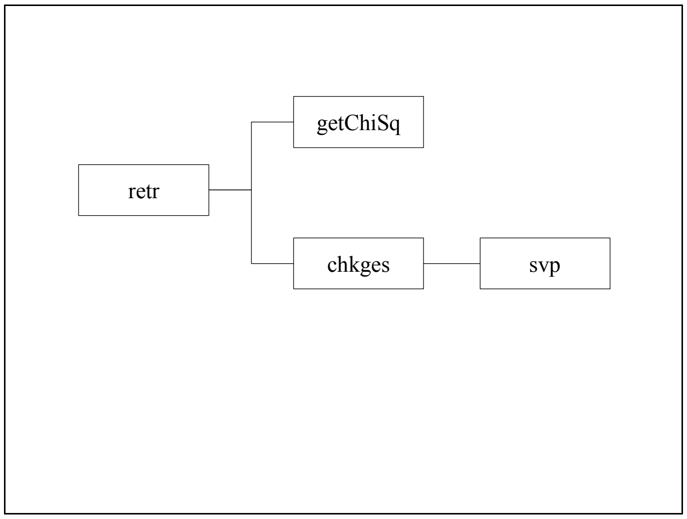

Figure 5: Software Components of Quality Control

Table 4: Functional Description of Quality Control

1st level		**Description**
getChiSq		Calculates χ^2 for quality control
chkges		Checks variables with the physical limits
	svp	Calculates saturation vapor pressure

2.5.5 Post-processing

This last module in the core retrieval algorithm is illustrated in Figure 6 and Table 5. The names of output files from the retrieval are contained in the control file to be discussed in Section 2.7, Table 8 and Figure 8).

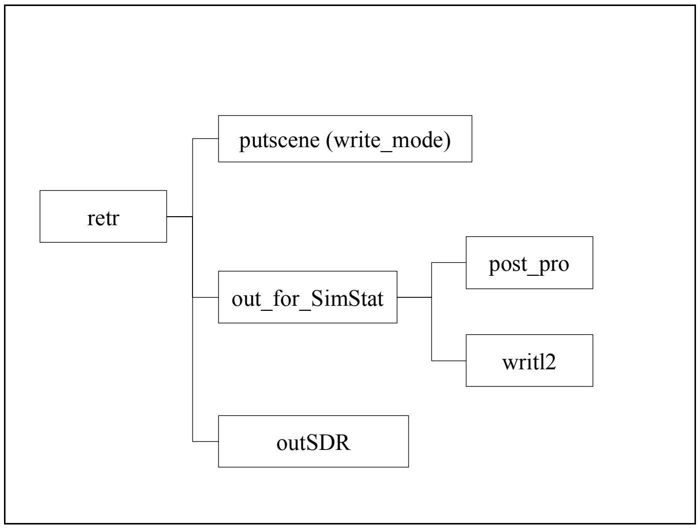

Figure 6: Software Components of Output and Post-processing

Table 5: Functional Description of Output and Post-processing

1st level	**2nd level**	**Description**
putscene (write_mode)		Outputs the retrieved parameters
out_for_simstat		Outputs EDRs in the IPO ASCII format
	post_pro	Interpolates OSS outputs to IPO format
	writl2	Writes variables according to IPO format
outSDR		Outputs SDRs

2.6 I/O Format

The main inputs and outputs to the retrieval algorithm are SDRs and EDRs, respectively. A structured data file format netCDF is used for the output files (the same format will eventually be adopted for the SDR files as well). Data records in this format are organized in a compact manner and can be accessed directly. In addition, files created in the netCDF format are independent of computing platforms. The netCDF library offers general methods for reading, writing, and direct accessing the record in the file. It also provides science teams with a robust interface to many commercial analysis tools. Another advantage of the netCDF format is that it is self-describing, with the global and variable attributes specified in the header of the file, followed by the variable vectors organized according to their attributes.

2.6.1 SDR

The current version of the software expects the SDR files to be in the IPO ASCII format. Ideally, however, the retrieval module should be able to handle a variety of formats from a variety of instruments (e.g., NESDIS). This requires a standard SDR interface that allows the individual data formats to evolve over time, without having any impact in the retrieval algorithm. The various formats would then be converted into a fixed AER format and if the original format were to change, only the converter program would have be upgraded and not the retrieval algorithm.

2.6.2 Core Module EDRs

The EDRs in the core retrieval module are represented as vectors, with all retrieved variables stacked into one vector. The vector is used at all stages of the retrieval. The vector has two different lengths, depending on whether it is in the geophysical (output) or the EOF (core retrieval) domain. The vector implementation simplifies the software design and is consistent with the theoretical basis of the maximum likelihood inversion algorithm. The contents of an EDR vector in the geophysical domain are described in Table 6, while the contents in the EOF (or retrieval) domain are described in Table 7. Figure 7 shows a portion of an EDR netCDF file in the geophysical domain.

Table 6: Vector Structure in the Geophysical Domain.

Description	**Starting Index**	**Symbol for th Starting inde**	**# of Element**	**Default # of Elements**
Temperature	1		NLev	40
Skin temperature	NLev+1	ITskinG	1	1
Surface Pressure	ITskinG+1	IPsfcG	1	1
Water vapor	IPsfcG +1	IH2oG	NLev	40
MW Cloud Parameter	IH2oG+NLev	ICldMwG	NCldMwG	3
MW Emissivity	ICldMwG+NCldMG	IEmMwG	NEmMwG	40
Total Length				125

Table 7: Mapping Structure in the Retrieval Domain.

Description	**Starting Index**	**Symbol for the Starting Index**	**# of Elements**	**Default # of Elements**
Temperature	1		Ntemp	20
Skin temperature	NTemp+1	ITskin	1	1
Surface Pressure	ITskin+1	IPsfc	1	1
Water vapor	IPsfc+1	IH2o	NH2O	10
MW Cloud Parameter	IH2o+NH2o	ICldM	NCldMw	3
MW Emissivity	ICldMw+NCldw	IEmisMw	NEmMw	12
Total Length				47

```
netcdf truth.scene {
dimensions:
        nPar = 125 ;
        nProfiles = UNLIMITED ; // (200 currently)
        nTime = 6 ;
        nId = 11 ;
variables:
        float profiles(nProfiles, nPar) ;
                profiles:long_name = "set of profile vectors" ;
        char id(nProfiles, nId) ;
                id:long_name = "Identification String Tag " ;
        float lat(nProfiles) ;
                lat:long_name = "latitude values " ;
                lat:units = "degrees north" ;
        float lon(nProfiles) ;
                lon:long_name = "longitude values" ;
                lon:units = "degrees east" ;
        int landType(nProfiles) ;
                landType:long_name = " type (land/ocean/etc)" ;
        float pland(nProfiles) ;
                pland:long_name = "% land (0..1)" ;
        float height(nProfiles) ;
                height:long_name = "satellite height" ;
                height:units = "km" ;
        float scanAngle(nProfiles) ;
                scanAngle:long_name = "satellite scan angle" ;
                scanAngle:units = "degrees" ;
        float date(nProfiles, nTime) ;
                date:long_name = "date and time" ;
                date:units = "years-months-days-hours-min-secs" ;

// global attributes:
                :Version = "AER-0001" ;
                :Mode = 1 ;
                :CreationDate = "Wed Sep  6 10:36:06 2000" ;
                :ModeDescription = "Geophysical Space" ;
                :Temperature = 0, 40 ;
                :Tskin = 40, 1 ;
                :SurfacePressure = 41, 1 ;
                :H2o = 42, 40 ;
                :MwCloud = 82, 3 ;
                :MwEmissivity = 85, 40 ;
                :IrCloud1 = 125, 0 ;
                :IrCloud2 = 125, 0 ;
                :IrEmissivity = 125, 0 ;
                :IrReflectivity = 125, 0 ;
                :Co2 = 125, 0 ;
                :O3 = 125, 1 ;
                :N2o = 125, 0 ;
                :Co = 125, 0 ;
                :Ch4 = 1125, 0 ;
                :StandardMWemis = 0 ;
                :standardPressureGrid = 0.1f, 0.2f, 0.5f, 1.f, 1.5f, 2.f, 3.f, 4.f, 5.f, 7.f, 10.f, 15.f, 20.f, 25.f, 30.f, 50.f, 60.f, 70.f, 85.f,
100.f, 115.f, 135.f, 150.f, 200.f, 250.f, 300.f, 350.f, 400.f, 430.f, 475.f, 500.f, 570.f, 620.f, 670.f, 700.f, 780.f, 850.f, 920.f, 950.f, 1000.f ;
                :mwSurfaceFrequencies = 23.8f, 31.4f, 50.3f, 52.8f, 53.596f, 54.4f, 54.94f, 55.5f, 57.2903f, 57.2903f, 57.2903f,
57.2903f, 57.2903f, 57.2903f, 89.f, 89.f, 150.f, 183.31f, 183.31f, 183.31f ;
                :mwSurfacePolarities = 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ;;
data:

 profiles =
  241.4208, 250.5571, 257.8849, 270.5988, 270.7998, 265.7951, 254.792,
......
 id =
  "IPR00001",
......
 lat = -21.922, -21.922, -21.977, -21.977, -39.398, -39.398, -43.523,
......
 lon = -137.633, -137.633, -159.617, -159.617, -11.18, -11.18, 148.789,
.......
 landType = 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17,
.......
 date =
  1988, 1, 1, 4, 0, 0,
.......
}
```

Figure 7: Example of a netCDF EDR file.

2.7 Control Parameters

The CMIS retrieval algorithm uses an ASCII control file to load the necessary parameters at runtime. This file contains a set of user-defined constants and I/O filenames used by the retrieval modules. These parameters are formulated in terms of FORTRAN-formatted namelists. Figure 8 shows an example of the control file and Table 8 contains the description of all the namelist fields. As discussed in the section 2.2, one of the desired features of the retrieval module is its flexibility. Consequently, the control parameters can be modified either directly in a control file (an example of the control file is given in ***test.1.in*** located in the ***Data/CalVal/Auxiliary*** directory) or by running the top-level Perl-language script, ***urfront***. The script file generates the corresponding control file (suchas ***test.1.in***) in the respective data directory, utilizing the configuration file ***etc/UR.conf*** and ***devices/cmis/etc/control.conf***, and the command line options. The list of options for the script command is shown in Table 9. For more information on using the script file, the user may refer to Section 3.3.

```
$stdInputs
   nfor=200
   nfov=1
   debug=.FALSE.
$
$stdMwInputs
  nxiter=9
   mwCld=1
   iCell=1
   PiCell=1
   icasc=1
   nchmwIn=40
  kchanmw=1,1,1,1,0,0,1,1,0,0,0,0,1,1,1,1,0,0,1,1,
                1,1,1,1,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1
  landTypes=1,2,3,4
$
$stdIrInputs
   mxiter=4
   iClssMode=3
   ozofac_bg=1.0
$
$stdInputFiles
  chanFileName='./Database/fileholder.dat'

$
$inputFiles
   scenefile='./Data/global/Scenes/truth.scene.nc'
   wptbProfFile='./Data/global/Scenes/true.scene.asc'
   wptbAuxFile='./Data/global/Auxiliary/aux-test.asc'
   wptbScanFile='./Data/global/Auxiliary/fileholder.dat'
$
$outputFiles
   mwRetrAscFile='./Data/global/EDRs/
                            retr.mw1.scene.1.asc'
   irRetrAscFile='./Data/global/EDRs/fileholder.dat'
   qcAscFile='./Data/global/Auxiliary/qc.asc'
   mw1SDRFile='./Data/global/SDRs/mw1SDR.1.asc'
   mw2SDRFile='./Data/global/SDRs/fileholder.dat'
   irSDRFile='./Data/global/SDRs/fileholder.dat'

mwRetrNcFile='./Data/global/EDRs/retr.mw1.scene.1.nc'
   guessNcFile='./Data/global/EDRs/guess.scene.1.nc'

retrMwrad='./Data/global/SDRs/retr.mw1.radiance.1.asc'
$
```

```
$sensorConf
   nednFile='./Database/nedn_710.dat.frq'
   freqFile='./Database/outfreq2.dat'
   noiseParFile='./Database/OSS/cmis/noise.dat'
   nedtFile='./Database/sensorModel/cmis/
                 nedt_0012_pdr.dat'
  noiseReductFactFile='./Database/sensorModel/cmis/
                            cmis_nrf_with_cal_pdr.dat'
  chan_parms='./Database/sensorModel/cmis/
                       cmis_aux_pdr0012.dat'
$
$inputConf

covarFile='./Database/Statistics/cmis/ocean.compresse
d.pdr.nc','./Database/Statistics/cmis/landseaice.compre
ssed.pdr.nc','./Database/Statistics/cmis/coldice.compre
ssed.pdr.nc','./Database/Statistics/cmis/landhiemis.co
mpressed.pdr.nc'
   emisfilg='./Database/guess.emissivity'
   mwrsk='./Database/OSS/cmis/coef_oss_rsk_pdr'
   mwopt='./Database/OSS/cmis/opdeptab_rsk_pdr'
   mwtbl='./Database/OSS/cmis/odgrid_pdr'

   mwrsk='DataBase/oss/C_v.rsk'
   mwopt='DataBase/oss/opdepth.oss.rsk'
   mwtbl='DataBase/oss/prof.tab.oss'
$
$cascadeIO
   cascinfile='./Data/global/Auxiliary/cascfileout.1'
   cascoutfile='./Data/global/Auxiliary/cascfileout.1'
 $
 $extraConf

 $
```

Figure 8: Example of the namelist control file.

Table 8: Parameters in the Control File for the Retrieval Module.

Namelist	**Parameter**	**Description**
stdInputs		Standard IR/MW inputs
	nfor	Number of FOR
	nfov	Number of FOV
	debug	On-screen output flag for simulation
stdMwInputs		MW specific inputs
	nxiter	Maximum number of iteration
	mwCld	MW cloud flag (0 = no cloud, 1 = cloud)
	iCell	FOR resolution flag (1 ~ 5 for 50 km ~ 15 km)
	Picell	previous cell #, used for cascade mode
	icasc	cascade flag
	nchmwIn	Number of MW channels used
	kchanmw	Vector of MW channel selection flags (0 = off, 1 = on)
	landTypes	Land type flag (1 = ocean, 2 = land, 3=, 4=,)
stdIrInputs		spared input namelist
	mxiter	spared parameter
	iClssMode	spared parameter
	ozofac_bg	spared parameter
stdInputFiles		spared namelist
	chanFileName	spared parameter
inputFiles		I/O files
	scenefile	Profile files for simulation in netCDF format
	wptbProfFile	Profile files in IPO ASCII format
	wptbAuxFile	Auxiliary data file
	wptbScanFile	Scan line test data file
outputFiles		Output files from the retrieval module
	mwRetrAscFile	CMIS sensor retrieval profiles in ASCII format
	irRetrAscFile	spared parameter
	qcAscFile	quality control file (off by default)
	mw1SDRFile	CMIS SDR
	mw2SDRFile	spared parameter
	irSDRFile	spared parameter
	mwRetrNcFile	MW-band retrieval profiles in netCDF format
	irRetrNcFile	IR-band retrieval profiles in netCDF format
	guessNcFile	background profiles in netCDF format
	retrMwRad	Radiance output in MW-band
sensorConf		Sensor specific configuration file
	nednFile	NEDN file
	freqFile	spared parameter
	noiseParFile	MW noise file
	nedtFile	NEDT file
	noiseReductFactFile	CMIS noise reduction factor file
inputConf		Inputs for OSS forward model
	covarFile	Background, covariance and eigenvector file

	emisfilg	spared parameter
	mwrsk	MW coefficient file
	mwopt	MW optical depth file
	mwtbl	MW layer averaged lookup table
cascadeIO		Cascade I/O files
	cascinfile	Input file for cascade mode
	cascOutfile	Output file for cascade mode

Table 9: Options for Top-level Script File “urfront.”

Options	**Types**	**Default (& option) values**	**Descriptions**
run	<string>	none	process name
nprofs	<int>	30	# FORs
dataPath	<string>	none	I/O data path
terrain	<string>	none ([land | ocean])	terrain type
devConf	<string>	devices/[sensor]/etc/control.conf	device configuration file
urConf	<string>	etc/UR.conf	common parameter file
runConf	<string>	$(dataPath)/Auxiliary/test.1.in	run-time configuration file
iCell	<int>	1	Cell size number for resolution
PiCell	<int>	1	previous cell # for cascade mode
icasc	<int>	0	cascade flag (0: none)
help		none	online help

2.8 Directory Structure

All files in the CMIS software are managed in a directory tree structure. Figure 9 shows the top-level directories and their descriptions once the software is installed. Specific to the deliverable software are the test profiles, SDRs, and the corresponding retrieved EDRs that can be used for validation purposes. Also included are the graphics tools written in IDL for post-processing and assessing the product and a freeware library of netCDF functions. The library allows the creation of more profiles for simulation purposes in the format specified in the ***DataBases/standardTestScene/*** sub-directory.

```
urfront-> command line interface
GNUmakefile -> Top directory makefile
makefile.common-> common makefile included by major processes
validation.csh-> C-shell script for validation runs
./bin-> CrIS executable binary files
./etc-> configuration files
./retr-> retrieval module
   ./src -> source code
   ./include -> include files
./common-> shared code by various modules
   ./src -> source code
   ./include -> include files
./Database-> static database
  /EDR-> inputs for prospective EDRs
  /OSS -> data for OSS IR and MW models
  /sensorModel->proposed sensor parameters
  /Statistics -> background and covariance files
./Data-> SDR
  /CalVal -> calibration/validation data
  /global-> global scenes with cloud
  /ocean_clear-> ocean, clear sky scenes
  /ocean_cloudy-> ocean, cloudy sky scenes
  /land_clear->land, clear scenes
  /land_cloudy->land, cloudy scenes
  /norfi-> no radiative frequency interference scenes
  /rfi10-> with radiative frequency interference at 10 GHz
  /rfi37-> with radiative frequency interference at 37 GHz
./lib-> class libraries
    /common -> common utility code
    /mat -> matrix manipulation code
    /ncdf -> netCDF library code
    /perl     ->  perl
./docs -> document for the code
```

Figure 9: Directory structure of the CMIS software.

3 TUTORIAL

This section provides a tutorial to an idealized test of the software of the core retrieval module. The independent EDR test may be referred to the command line help or the companion top-level C-shell script file, validation.csh.

In this test, SDRs are given based upon known “scenes” (i.e., temperature and moisture profiles, surface emissivity, etc.) and combined with auxiliary data (latitude and longitude, satellite angle and height, and date/time). In a “true” end-to-end test of the CMIS algorithm, additional factors, such as instrument and spectroscopy errors, should be taken into account.

The SDRs and statistics (climatological background, error covariance matrix, and eigenvectors) are stored in the *Database* directory. They are used by the retrieval algorithm to generate the geophysical parameters at internal pressure levels. The algorithm also outputs intermediate products such as cloud-cleared radiances for post-processing purposes. All EDRs are located in one directory as specified by the option -dataPath at ***urfront*** command line (see description of ***urfront*** in Section 3.3).

The post-processing and all the EDR modules takes the geophysical parameters retrieved by the core retrieval module and generates the final products according to the requirements. The final products are in netCDF format and located in the same directory as specified by –dataPath.

3.1 Install the Software

The software and the associated data inputs are in two tar files delivered on two CDs. To install the software on a designated platform (preferably an SGI machine), the user should perform the following steps:

Step 1. Insert the CD on a local or network CD drive.
Step 2. Type: <CDROM_PATH>/Install.
where the <CDROM_PATH> stands for the path of which the CDROM is mounted.

In adition to the tar file, the CD also contains two other files: a README file and an installation script file. Upon running the installation script file, the tar file will be expanded and a directory tree shown in Figure 9 will be created in the subdirectory CMIS_1001 for the first CD and CMIS_1002 for the second CD.

3.2 Build the Software

The makefiles at each level of the software are written in GNUMakefile format. After changing the directory to CMIS_100, at the UNIX command line, type: ***gmake build***. This will build the executables for the retrieval process and the individual EDRs in the ***bin/*** directory.

3.3 Top-level Command "*urfront*"

The easiest way to run the CMIS retrieval algorithm is by using the top-level command ***urfront***. This command allows to run the retrieval with various run options. The user can change the run options interactively at the runtime by entering command line arguments. The options can also be defined in a configuration file. Typically, the user maintains a set of configuration files for different scenarios. After each run, ***urfront*** saves all configuration information in the Auxiliary directory under the –dataPath. This facilitates the rerun of same scenarios.

3.3.1 Running *urfront* with command line options

A list of currently available options for urfront is given in Table 9 (it should be noted that ***urfront*** is designed to accomodate the addition of new command line options). Among the options most often used in the current version of the software are the following:

-help

Prints the available set of options, their types and current values.

-run <process>

Tells ***urfront*** which component of the system to run, with <**process**> equal to 'sim' or 'retr.' For example,

```
 % urfront -run retr
```

runs the core retrieval module.

-urConf <filename>

This option selects the configuration file, which contains general variable names and values for running the CMIS retrieval. The default value is *etc/UR.conf*. This file contains all the sensor-associated parameters for running the retrieval.

--runConf <filename>

This argument specifies which device specific configuration file to use. This configuration file contains parameters specific to the sensor. Values defined in this file override values defined in any of the upper level configuration `file(s)`. In general, the latter variable definition takes precedence over former variable definition, and the command line argument overrides any variable definition made in configuration files.

3.3.2 Running *urfront* using configuration file

urfront uses the so-called configuration file (default: *etc/UR.conf* and *devices/cmis/etc/control.conf*) and converts it into a FORTRAN namelist file to be used for running the software (see Section 2.7). In this way, the software can be run with different internal parameters and different I/O files without the need to recompile the FORTRAN code. The configuration file holds information to configure the CMIS retrieval with sensor characteristics, inputs, and outputs at runtime. The user can easily modify the default configuration file. An example of using ***urfront*** with the default configuration file is shown in Section 3.4.

3.3.3 Structure of the configuration file

Configuration Lines

A configuration file has three types of lines: comments, variable definitions and name lines for namelists.

A comment line starts with a pound sign #. Comment lines are skipped by **urfront** when generating the master control file but they are useful for documenting variables and maintaining configuration files.

A variable definition line has the format **name=value**. This is a convenient way for defining a symbolic name that one can apply repeatedly to the definition of configuration values within namelist blocks.

A namelist starts with a dollar sign **$** and ends just before another namelist. Within each namelist block, there may be many variable definitions that correspond to the setup in the FORTRAN code.

Example of configuration file

In this configuration we give to the symbol name DATA_DIR the value */npoess/test/data*.

```
# in the header section we can define variables that will be used
# repeatedly in the definition of namelists.

DATA_DIR=./Database/OSS/cmis
# now you can use pre-defined symbol to define variables that
# configure the CMIS retrieval.

# the namelist
$inputConf
mwrsk='$(Database)/coef_oss_rsk_pdr'
mwopt='$(Database)/opdeptab_rsk_pdr'
mwtbl='$(Database)/odgrid_pdr'


# The urfront will replace the expression $(DATA_DIR) with
# ./Database. The definition of such symbolic names
# facilitates the maintenance of configuration files.
```

The resulting namelist would be:

```
$inputConf
mwrsk = './Database/OSS/cmis/coef_oss_rsk_pdr'
mwopt = './Database/OSS/cmis/opdeptab_rsk_pdr'
mwtbl = './Database/OSS/cmis/odgrid_pdr'
$
```

Configuration files have two sections, a header section and a namelist section:

Header Section

The header section is the first block in a configuration file and is used for variable definitions. These variables could be used in the formation of namelists or could be used directly by the ***urfront*** program. The header section ends where the first namelist block starts. This section does not make part of the namelist. This section facilitates the management of the CMIS complex configuration parameters.

Namelist Section

Namelist blocks have the similar form of the FORTRAN namelist that will be copied and/or modified by ***urfront*** to form a master control file. Each namelist holds one or more CMIS configuration parameters and has its corresponding definition in the FORTRAN code. If new variables are added to a namelist or a new namelist is created, the FORTRAN code has to be modifed to reflect these changes in the configuration file.

Each namelist in the control file will be passed to the FORTRAN code. The name part corresponds to a variable name in the FORTRAN code. The value part corresponds to a specific FORTRAN data type. The ***urfront*** interface either applies values defined in the header section for particular names or overwrites with command line values for defining namelists.

The function reading configuration information expects a predefined set of namelists and variables.

3.4 Retrieval

This is the core process of the CMIS software. The process retrieves the designated atmospheric parameters and outputs them in the same format as the scene profiles. The retrieval process has its specific parameters specified in the retr/include/ directory (in addition to those defined in common/include, and devices/cmis/include/retr). To invoke the retrieval process, at the commands line type:

```
 urfront –run retr –dataPath Data/global –nprofs 5
```

where the "-run" option is for the prospective executable; The "-dataPath" is for the specified dataset. The outputs are in the directory, Data/global/EDRs. The number of profiles is five.

The retrieval code can use either the full set of CMIS channels or a subset selected based on the method for graceful degradation. The default channel set used in the code is specified in the namelist file. The retrieval can also be performed by running the executable file directly:

```
bin/cmis < Data/global/Auxiliary/test.1.in
```

3.5 EDRs

The commands for executing individual EDRs are discussed in Volume III of this documentation.

3.6 Validation Batch File

In order to validate the performance of the software under various circumstances, a script file ***validation.csh*** is provided for running a series of cases in the batch mode. The script can be used for all cases or selected cases, by commenting or adding certain commands in the file. To run the batch file, the user should type ***validation.csh*** at the shell prompt. The user may refer to the header of the script file for more information

3.7 Useful Utility

The I/O files are mainly in the netCDF format and require additional for analyzing them. A "quick look" at a netCDF file is provided by the command is ***ncdump***, which has the syntax

```
ncdump <netCDF file> | more
```

This page intentionally left blank.